# Résolution numérique en Python

WANG Jinxin 3404759

December 3, 2014

**Abstract**

1) Méthode de résolution numérique dans les systèmes dynamiques
2) Équation de diffusion

# 1 Méthode de résolution numérique dans les systèmes dynamiques

## 1.1 Compléter (3) afin que le code implémente la méthode d'Euler

```
for i in range(n-1):
    x[i+1] = x[i] + h * func(x[i]) # (3)
```

## 1.2 En compléter (1) et (2), résoudre numériquement l'équation de croissance logistique

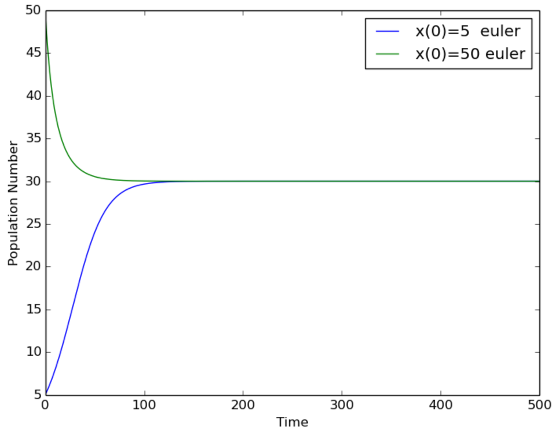
```
def f(x):
    return 0.1 * x * ( 30 - x ) # (1)

def init_x(val_init_x, n):
    if type(val_init_x) == type([]):
        x = np.zeros([n,len(val_init_x)])
    else:
        x = np.zeros([n,])
    return x

def euler(func, val_init_x, tf = 100, n = 500):
    h = tf/float(n)
    x = init_x(val_init_x, n)
    x[0] = val_init_x # (2)
    for i in range(n-1):
        x[i+1] = x[i] + h * func(x[i])
    return x

euler(f, 5, tf=10)
euler(f,50, tf=10)
```

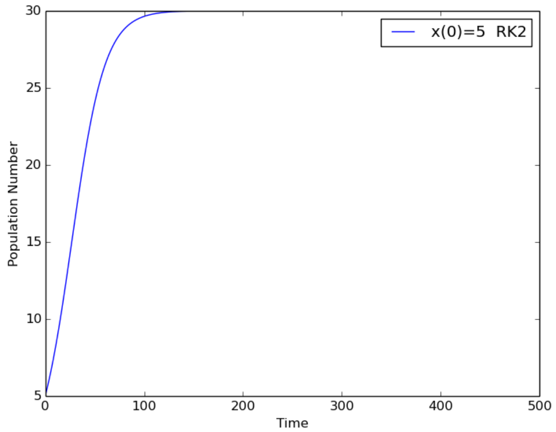
## 1.3 DM Tracer sur un meme graphe l'équation de la popluation poiur x0=5 et x0=50




## 1.4 Runge-Kutta

```
def RK2(func, val_init_x, tf = 100, n = 500):
    h = tf/float(n)
    x = init_x(val_init_x, n)
    x[0] = val_init_x
    for i in range(n-1):
        x[i+1] = x[i] + (h/2) * func(x[i])
                 + (h/2) * func(x[i]+h*func(x[i]))
    return x

RK2(f, 5, tf=10)
```



## 1.5 Un programme qui évqlue le moyenne du carré de l'erreur

```
def calcul_erreur_Q6(func, val_init_x, tf = 100, n = 500):
    return ((sum(euler(func, val_init_x, tf, n))
            - sum(RK2(func, val_init_x, tf, n)))**2)/n
```

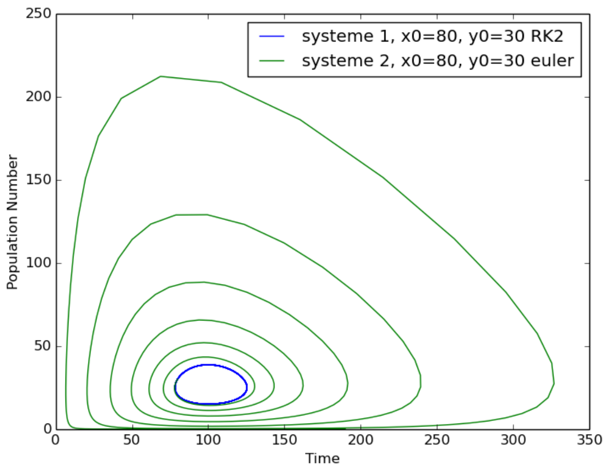
## 1.6 En utilisant la méthode d'Euler, modifier le code pour résoudre le système

```
def f2(variable):
    [ x, y ] = variable
    return np.array([0.25*x - 0.01*x*y, 0.01*x*y - y])

def init_x(val_init_x, n):
    if type(val_init_x) == type([]):
        x = np.zeros([n,len(val_init_x)])
    else:
        x = np.zeros([n,])
    return x

euler(f2,[80,30])
```

## 1.7 Tracer le portrait de phase pour x(0) = 80 et y(0) = 30




```
points = RK2(f2,[80,30])
plt.plot(points[:,0], points[:,1], label='systeme 1, x0=80, y0=30 RK2')
points = euler(f2,[80,30])
plt.plot(points[:,0], points[:,1], label='systeme 2, x0=80, y0=30 euler')
```

### 1.8 La solution obtenue correspond-t-elle au résultat attendu

Le figure obtenu correspond au résultat attendu. En temps passe, la popluation circule et approximate. Ils sont en fin stables.

## 2 Équation de diffusion

### 2.1 simuler l'évolution de la diffusion

```
# ---- Exercise 2 Equation de diffusion ----
import numpy as np
import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter

tf = 100; nt=100; nx=100; a=1e-3; x=10
dt = tf/float(nt)
dx = x/float(nx)
x0 = nx/2

U  = np.zeros([nt,nx])
U[0, x0] = 1

for k in range(0, nt-1):
    for i in range(1,nx-1):
        U[k+1,i]= U[k,i]+a*U[k,i+1]-2*U[k,i]+U[k,i-1] # (4)

fig = plt.figure()
ax  = fig.gca(projection ='3d')
X   = np.arange(0, x, dx); T = np.arange(0,tf,dt)
X, T= np.meshgrid(X,T)
surf= ax.plot_surface(X, T, U, rstride=1, cstride=1, cmap=cm.coolwarm, linewidt
fig.colorbar(surf)
plt.show()
```

### 2.2 La condition de bord dans le programme précedent correspond à 1